

```

/*-----*
 * File Name: CurveSplineInterpolate.c *
 * Creation: ER, 01/27/05 *
 * Purpose: Programming Example *
 * Copyright (c) OriginLab Corp.2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010 *
 * All Rights Reserved *
 * *
 * Modification Log: *
 *-----*/

#include <Origin.h>
#include <OC_nag8.h> // needed for nag_ functions

////////////////////////////////////
// This example shows how to perform 1D spline interpolation of a curve by
// calling NAG library functions.
// Call this function with a graph active that contains a data plot,
// or with a worksheet active that has a Y column selected.
//
// Parameters:
//     nPts:   Number of points desired in the spline curve
//             Default value is 100
//
void curve spline interpolate(int nPts = 100)
{
    // Get the active curve in the project and check validity.
    // Active curve can be curve in a data plot, or a Y column selected in a wks
    using crv = Project.ActiveCurve();
    if( !crv )
    {
        out_str("Could not find a valid active curve!");
        return;
    }

    // Get the corresponding X dataset and check validity
    Dataset dsX;
    BOOL bRet = crv.AttachX(dsX);
    if( !bRet )
    {
        out_str("Failed to find associated X dataset of active curve!");
        return;
    }

    // Get names of X and Y datasets associated with the curve.
    // The Y dataset name is the name of the curve itself.
    string strYName = crv.GetName();
    string strXName = dsX.GetName();

    // Get curve size
    int nSize = crv.GetSize();
    // Check number of points asked for in spline curve.
    // This checking is arbitrary - can be changed as desired.
    if( nPts < nSize || nPts > 10 * nSize)
    {
        out_str("Invalid number of points asked for in the result spline curve!");
        return;
    }

    // Copy the X and Y datasets of the curve to vectors.
    // Cannot pass datasets directly to NAG function.
    vector vecX, vecY;
    // Y values are in the curve object itself.
    vecY = crv;
    vecX = dsX;

    // Declare structure to hold NAG spline coefficients and call NAG function
    Nag_Spline spline;
    Nag_Error errInterpolant;
    nag_1d_spline_interpolant(nSize, vecX, vecY, &spline, &errInterpolant);
    // If no error, then proceed to generate a table of values using spline coefficients
    if( NE_NOERROR == errInterpolant.code )
    {
        // Create a new worksheet
        Worksheet wksSpline;
        wksSpline.Create();
        // Get the page object and set its label, and turn on display of label
        WorksheetPage wpg = wksSpline.GetPage();
        wpg.Label = "Spline interpolation for curve (" + strXName + ", " + strYName + ")";
        wpg.TitleShow = WIN_TITLE_SHOW_BOTH;
        // Delete all existing columns and add two new columns
        while( wksSpline.DeleteCol(0) );
        wksSpline.AddCol();
        wksSpline.AddCol();
        // Set 1st column to be of type X
    }
}

```

```

wksSpline.Columns(0).SetType(OKDATAOBJ DESIGNATION X);
// Declare datasets corresponding to these two columns
// These two datasets will hold X, Y values of the spline curve to be generated
Dataset dsXSpline(wksSpline, 0);
Dataset dsYSpline(wksSpline, 1);
// Fill the X dataset with nPts from begin to end values of the data curve
double dxStep = (dsX[nSize - 1] - dsX[0]) / (nPts - 1);
dsXSpline.Data(dsX[0], dsX[nSize - 1], dxStep);
// Set the size of the Y dataset to be same as X
dsYSpline.SetSize(nPts);
// Loop over nPts and compute spline Y values for given X value
for(int ii = 0; ii < nPts; ii++)
{
    double dx, dY;
    dx = dsXSpline[ii];
    NagError errEvaluate;
    nag_ld_spline_evaluate(dx, &dY, &spline, &errEvaluate);
    // If generation of spline Y value failed, report to user
    if( NE NOERROR != errEvaluate.code )
    {
        printf("Failed to generate spline y value at x = %f\n, NAG Error code = ",
            dx, errEvaluate.code);
        dY = NANUM;
    }
    dsYSpline[ii] = dY;
}
// All done with computing spline curve
return;
}
// If NAG returned error in creating spline coeffs from data curve, report appropriate message
else
switch( errInterpolant.code )
{
    case 11:
        out_str("Data size should be greater than 4.");
        break;
    case 63:
        out_str("X values need to be strictly increasing. Sort data and call again.");
        break;
    case 73:
        out_str("Memory allocation failed.");
        break;
    default:
        out_str("NAG function returned unknown error.");
        break;
}
}
////////////////////////////////////

```